



**Intermec**



User's Guide



**IF61 ALE Engine**

Intermec Technologies Corporation

Worldwide Headquarters

6001 36th Ave.W.

Everett, WA 98203

U.S.A.

[www.intermec.com](http://www.intermec.com)

The information contained herein is provided solely for the purpose of allowing customers to operate and service Intermec-manufactured equipment and is not to be released, reproduced, or used for any other purpose without written permission of Intermec Technologies Corporation.

Information and specifications contained in this document are subject to change without prior notice and do not represent a commitment on the part of Intermec Technologies Corporation.

© 2007-2008 by Intermec Technologies Corporation. All rights reserved.

The word Intermec, the Intermec logo, Norand, ArciTech, Beverage Routebook, CrossBar, dcBrowser, Duratherm, EasyADC, EasyCoder, EasySet, Fingerprint, i-gistics, INCA (under license), Intellitag, Intellitag Gen2, JANUS, LabelShop, MobileLAN, Picolink, Ready-to-Work, RoutePower, Sabre, ScanPlus, ShopScan, Smart Mobile Computing, SmartSystems, TE 2000, Trakker Antares, and Vista Powered are either trademarks or registered trademarks of Intermec Technologies Corporation.

Part of the software embedded in this product is gSOAP software.

Portions created by gSOAP are Copyright (C) 2001-2004 Robert A. van Engelen, Genivia inc. All Rights Reserved.

The software in this product was in part provided by Genivia Inc. Any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

ACE™, TAO™, CIAO™, and CoSMIC™ (henceforth referred to as “DOC software”) are copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright (c) 1993-2006.

There are U.S. and foreign patents as well as U.S. and foreign patents pending.

# Contents

About This Guide . . . . .	1
About the ALE Standard . . . . .	1
About the IF61 ALE Engine . . . . .	1
Installing or Upgrading the ALE Engine . . . . .	2
Starting, Stopping, and Managing the ALE Engine . . . . .	2
Configuring the ALE Engine . . . . .	3
About the ALE Configuration File . . . . .	3
About TriggerSetupCommand . . . . .	6
Changing the ALE Configuration File . . . . .	6
Intermec ALE-Specific Implementation of ALE 1.1 . . . . .	7



## About This Guide

This User's Guide explains how to configure and use the Application Level Events (ALE) Engine edgware application on the IF61 Fixed Reader. This Guide is intended for the person who develops RFID applications using the ALE Engine. You should already be familiar with the details of the EPCglobal ALE 1.1 specification.

For more information on operating the IF61, see the *IF61 Fixed Reader User's Manual* (P/N 935-011-xxx).

## About the ALE Standard

The EPCglobal ALE standard specifies methods by which large volumes of tag data can be processed into “events” that applications can easily manage. Processing can include reading tag IDs from multiple data sources, reporting data, accumulating data over a specified time period, filtering data to eliminate duplication, counting and grouping tag IDs to reduce total data volume, and writing to tags.

Version 1.1 of the ALE specification provides:

- read and write access to EPC data as well as other data present on EPC data carriers.
- easy expansion to include other tag types.
- specific operations such as “lock” and “kill” for Gen 2 tags.
- additional interfaces for defining tag memory fields, naming data sources, and securing the use of APIs.

For more information, see [www.epcglobalinc.org/standards](http://www.epcglobalinc.org/standards).

## About the IF61 ALE Engine

The IF61 Fixed Reader (with firmware version 2.10 or later) includes the ALE Engine, which is Intermec's implementation of the EPCglobal ALE 1.1 middleware specification. The default installation directory for the IF61 ALE Engine is `/home/developer/edgware/ale`.

Some features defined in the ALE 1.1 specification are optional or implementation-dependent. For information on how these features are supported by the IF61 ALE Engine, see “[Intermec ALE-Specific Implementation of ALE 1.1](#)” on page 7.

## Installing or Upgrading the ALE Engine

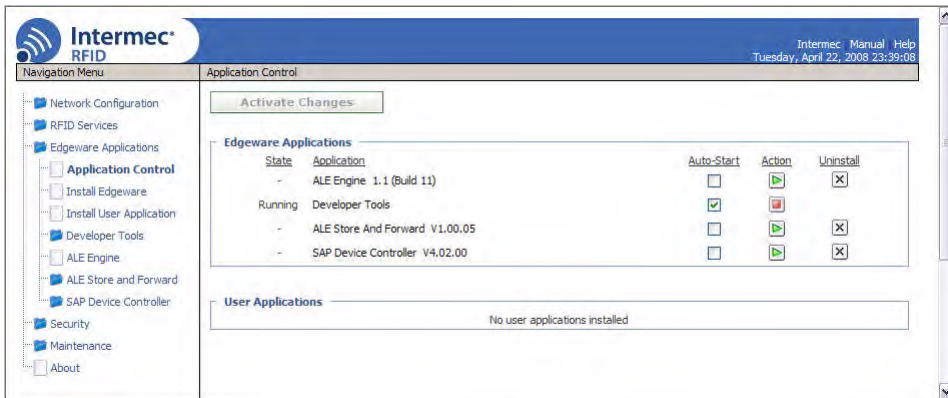
Use the IF61 web browser interface to install or upgrade ALE Engine firmware. For more information on upgrading, installing, and uninstalling the ALE Engine, see the IF61 user’s manual.

## Starting, Stopping, and Managing the ALE Engine

You can configure the ALE Engine to auto-start when the IF61 boots, or you can start and stop the ALE Engine manually. Follow the next procedure to start, stop, or manage the ALE Engine.



### To start or stop the ALE Engine

- 1 Open the IF61 web browser interface.
- 2 In the menu, choose **Edgware Applications**. The Application Control screen appears.



- 3 To run the ALE Engine whenever the IF61 boots, check the **Auto-Start** check box for the ALE Engine, and then click **Activate Changes**. The engine will automatically start the next time the IF61 boots.

You can also manually start or stop the ALE Engine as follows:

- If the ALE Engine is running, click  to stop it.
- If the ALE Engine is not currently running, click  to start it.

## Configuring the ALE Engine

On the IF61, the default logical reader is specified as:

`LocalAntennan`

where  $n$  is the number of the IF61 RFID antenna port. For example, “LocalAntenna1” is the antenna connected to RFID antenna port 1. All readers default to a polling value of 1 second.

The ALE Engine is preconfigured to define each of the four antenna ports as individual logical readers. External BRI-compatible devices, such as other IF61s, can be configured by creating new “PhysicalReader” sections for each of its antennas, and then assigning a unique name and IP address to the reader. Each of the newly defined physical readers can then be assigned to a pre-existing “LogicalReader” section, or assigned to a newly created “LogicalReader” section.

The default configuration file is `ale.conf`. All settings are case sensitive.

## About the ALE Configuration File

The `ale.conf` file is placed in the `/home/developer/edgware/ale/conf` directory and can be edited directly through the web browser interface page. For more information, see [“Changing the ALE Configuration File” on page 6](#).

Host applications can also use file transfer mechanisms such as FTP to update this file remotely.

The configuration file is as follows:

```
[config]
;IP Port number to which ALE listens
ALE_ServerPort=8512

;Maximum number of characters in any
;ALE identifier (ECSpec Name, ReaderName, etc)
ALE_MaxIdentifierLength=128
```

```

;Number of threads servicing incoming ALE requests
ALE_RequestServerThreads=10

;Number of threads servicing outgoing ALE notifications
ALE_NotificationThreads=10

;ALE Version string returned by the ALE GetStandardVersion() API
ALE_Version="1.0"

;Version string returned by the ALE GetVendorVersion() API
ALE_VendorVersion=""

;ALE ID string returned in ECRports
ALE_ID="Intermec ALE"

;String prefix for temporary ECSpec created for Immediate()
;and Poll() API calls
; Note: an integer value will be appended to this string to form a
; unique identifier for the temporary ECSpec.
ALE_ImmediateSpecName="___INTERMEC___IMMEDIATE___"

;ALE schema version returned in all ALE responses
ALE_SchemaVersion="1.0"

;Maximum backlog (in number of bytes)of unprocessed reader reports
ALE_MaxQueueSize=5000000

;Maximum number of unprocessed reader reports (per reader)
ALE_QueueThreshold=20

;Number of seconds ALE waits for a connection when sending an
;ECRports notification
SOAP_ConnectTimeout=5

;Number of seconds ALE waits for a send when sending an ECRports
;Notification
SOAP_SendTimeout=5

;Number of seconds ALE waits for a response when sending an ECRports
;Notification
SOAP_RecvTimeout=5

;Logical Reader definitions
; Each definition contains the physical readers that comprise the
; Logical Reader
; Multiple physical readers can be specified under each logical
; reader

[LogicalReaders\LocalAntenna1]
physAntenna1="First Antenna"

```



```

[LogicalReaders\LocalAntenna2]
physAntenna2="Second Antenna"

[LogicalReaders\LocalAntenna3]
physAntenna3="Third Antenna"

[LogicalReaders\LocalAntenna4]
physAntenna4="Fourth Antenna"

;Physical Reader Definitions
; ReaderType=BRI
;   --BRI is the only reader type currently
;       supported by Intermec ALE Engine
;   Address="ipaddr:port"
;   -- IP address (or hostname) and IP port
;       of the BRI port on this reader
;   Command="..."
;   -- BRI command for reading on this physical antenna
;   PollingInterval=integer_milliseconds
;   --Integer number of milliseconds between
;       polling cycles
;   TriggerSetupCommand=bri_command(s)
;   --Semi-colon separated BRI commands, usually used
;       to create triggers
[PhysicalReaders\physAntenna1]
ReaderType=BRI
Address="localhost:2189"
Command="attrib ants=1"
PollingInterval=1000
TriggerSetupCommand=TRIGGER "NAME1" GPIO 1 1 FILTER 5000;TRIGGER
"NAME2" GPIO 1 1 FILTER 5000

[PhysicalReaders\physAntenna2]
ReaderType=BRI
Address="localhost:2189"
Command="attrib ants=2;READ"
PollingInterval=1000

[PhysicalReaders\physAntenna3]
ReaderType=BRI
Address="localhost:2189"
PollingInterval=1000
Command="attrib ants=3;READ"

[PhysicalReaders\physAntenna4]
ReaderType=BRI
Address="localhost:2189"
PollingInterval=1000
Command="attrib ants=4;READ"

```

## About TriggerSetupCommand

You use TriggerSetupCommand to provide BRI commands that create named triggers. As with any BRI command, a semicolon separator allows you to define multiple BRI commands. No validation is performed on the provided commands, so you can use any valid BRI command. However, the commands are executed in a separate BRI session, so only commands that affect the global BRI state will have any effect on ALE behavior.

After you define the triggers, you can reference them in EC/CC Specs with the following URI syntax:

```
//intermec.com/trigger:xxxx
```

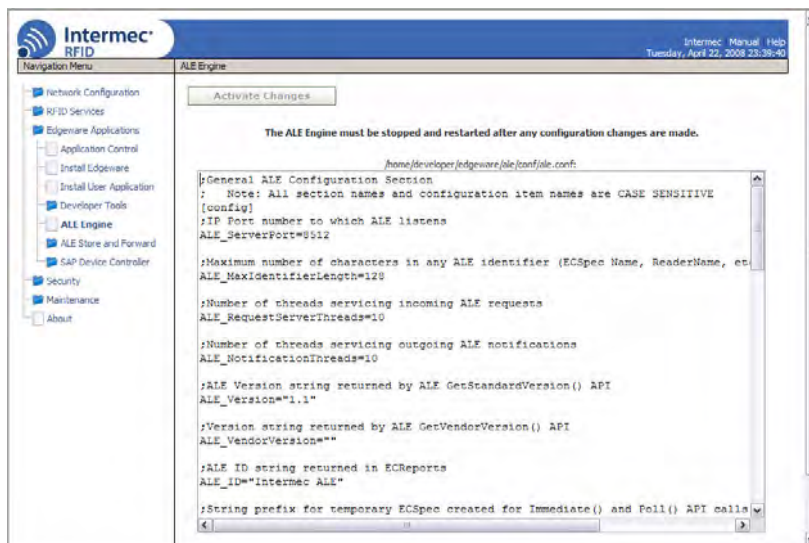
where *xxxx* is the case-sensitive name created by the BRI TRIGGER command.

## Changing the ALE Configuration File

You can access and make changes to the ALE configuration file from the web browser interface as described next.

### To edit the ALE configuration file

- 1 In the IF61 web browser interface, click **Edgware Applications > ALE Engine**. The ALE Engine screen appears.



- 2 Click in the text editor and make changes as necessary.
- 3 Click **Activate Changes**. Your changes are saved and will be made active the next time the ALE Engine is restarted.

If the ALE Engine is currently running and you want to make the changes active immediately, you need to manually stop and restart the ALE Engine. For help, see [“Starting, Stopping, and Managing the ALE Engine” on page 2](#).

## Intermec ALE-Specific Implementation of ALE 1.1

This section explains how the Intermec ALE supports optional or implementation-dependent features of the ALE 1.1 specification. Section numbers listed here are from the ALE specification.

### ALE Error and Status Logging

Intermec ALE logs operational information to the IF61 System Log. If you have problems when launching the ALE Engine, see the System Log for more information. For help with viewing the log, see the IF61 user’s manual.

### Section 4.3: Version Introspection Methods

Intermec ALE returns the empty string for all `getVendorVersion()` calls. Intermec ALE does not implement any vendor extensions and therefore returns the empty string.

### Section 4.5: Interpretation of Names

As a baseline, the specification requires that all implementations must accept non-empty strings that do not contain white space (Unicode `PATTERN_WHITE_SPACE`) or certain other “special” characters (Unicode `PATTERN_SYNTAX`).

As the specification allows, Intermec ALE expands on this by allowing white space inside identifiers, but no leading or trailing white space, and does not limit the use of any special characters except for the equals sign (=) or the vertical bar (|). All other characters are allowed.

Intermec ALE treats all identifiers as case-sensitive. For example, “fieldName” is not the same as “fieldname” or “FieldName”.

## **Section 6.1: Built-in Fieldnames**

Several built-in fields assume the capability of reading a tag bank of unknown length. The specification allows implementations that cannot read such a tag bank to raise an “operation not possible” condition if a client application attempts to use any of those fields.

Intermec ALE raises an “operation not possible” condition for the “epcBank” and “userBank” fields in the Reading API. As the specification requires, EC Specs containing references to these fields are accepted, but the fields are reported as “null” fields in any subsequent ECRpt. Additionally, as specified, the use of these fields in a primary key or filter causes the entire tag to be omitted.

### **Section 6.1.9: Generic Fieldnames**

In several places in the specification, the behavior of ALE with regard to EPCglobal Gen1 and Gen2 tags is documented along with a mandate that ALE behavior with “other” tag types be documented by the implementation.

Intermec ALE only supports Gen2 tags. The behavior with regard to Gen1 (and “other”) tags is undefined and untested.

#### **Section 6.1.9.2: Variable Fieldnames**

The specification requires that implementations be able to parse variable field definitions and accept EC/CC Specs that contain references to them. However, if an implementation does not support variable field operations, the specification allows the implementation to raise an “operation not possible” condition when actually interacting with tags.

Intermec ALE does not support read/write operations on tags for variable length fields.

#### **Section 6.2.1.3: EPC Datatype Pattern Syntax**

Intermec ALE does not support the “epc-pure” pattern syntax. Only the “epc-tag” pattern syntax is supported.

#### **Section 6.2.1.4: EPC Datatype Grouping Pattern Syntax**

Intermec ALE does not support the “epc-pure” grouping pattern syntax. Only the “epc-tag” grouping pattern syntax is supported.

### **Section 6.2.3.1: Binary Encoding and Decoding of the Bits Datatype**

According to Table 22 (Rules for Writing bits Values to Fields of Differing Lengths) from the ALE 1.1 specification:

- For  $M > N$ :  
The bits value to be written is longer than the available number of bits, so an “out of range” condition SHALL be raised.
- For  $M = N$ :  
The lengths match exactly; the value SHALL be written without modification.
- For  $M < N$ :  
The field is longer than the value. The value SHALL be written to the leftmost  $M$  bits of the destination. That is, the most significant bit of the value shall be aligned with the most significant bit position of the field. The remaining  $N-M$  bits SHALL each either be set to 0 or retain their previous value, at the discretion of the implementation.

For the “ $M < N$ ” case, the ALE specification leaves the bit padding behavior to the discretion of the implementation. For fields of a known length, Intermec ALE sets the remaining bits in the field to 0. For fields of unknown length (epcBank and userBank for example), Intermec ALE leaves the remaining bits unchanged.

### **Section 7: Tag Memory Specification API**

All features of the Tag Memory API have been implemented.

### **Section 8: ALE Reading API**

Intermec ALE is implemented only as a WS-I compliant SOAP binding. No other interfaces are supported.

Intermec ALE supports “http” and “tcp” reporting callback interfaces using XML. No other reporting methods are supported.

### Section 8.2.4: ECTrigger

The ECTrigger data element allows for vendor-specific start and stop triggers. In addition to the standard ALE triggers (such as the RTC trigger), Intermec ALE supports named BRI triggers. The triggers are created using the “TriggerSetupCommand” setting in the “ale.conf” configuration file. The named triggers are referenced in the ECTrigger data element using this syntax:

```
//intermec.com/trigger:xxxx
```

where xxxx is the case-sensitive name of the BRI trigger.

For more information, see [“About TriggerSetupCommand” on page 6](#).

#### Section 8.2.4.1: Real-time Clock Standardized Trigger

The specification makes it optional to implement this trigger type. Intermec ALE fully implements the Real-time Clock Standardized Trigger as specified.

The ALE V1.1 specification makes the following note for Real-time Clock Triggers:

“Note that if the specified period does not divide evenly into the number of milliseconds in a day (86,400,000), then the trigger will not be perfectly periodic, because the pattern will be realigned to the specified offset each day at midnight.”

Since each RTC trigger can specify a different time zone, there is an ambiguity as to when such triggers are realigned. Intermec ALE realigns any such triggers at midnight in the time zone referenced by the trigger in its definition – not at midnight according to the local time zone of the IF61.

### Section 8.2.13: ECStatProfileName

ECStatProfileName implementation is optional. Intermec ALE does not support ECStatProfileName.

### Section 8.3.9: ECTagStat

All of the ECTagStat (and its related data items and sub-classes) features are optional. Intermec ALE does not implement any ECTagStat related features.

### Section 9.3.5: CCOpType

Since ALE does not currently support ISO15962 encoded fields (variable length fields), the following CCOpType values are not allowed in CCSpecs, and will result in a CCSpecValidationException exception:

- “CHECK”
- “INITIALIZE”

In addition, the following CCOpType values will cause a CCSpecValidationException exception if they reference a variable length field:

- “ADD”
- “DELETE”

### **Section 9.3.9: CCStatProfileName**

Intermec ALE does not support CCStatProfileName values.

### **Section 9.4.4: CTagReport**

The ALE specification is silent about behavior after an error. Intermec ALE stops processing the CCmdSpec for a tag as soon as any error occurs (reading or writing). CCOpReport instances are reported for all operations up through the errant operation.

### **Section 9.5: EPCCache**

Many of the EPCCache-related methods return an EPCPatternList representing the EPC values contained in the cache before the method is performed. Unfortunately, this can easily lead to millions of EPC values being returned. Intermec ALE returns only the EPC patterns in the cache – not the realization of the individual EPC values represented by those patterns. For instance, a pattern such as “urn:epc:pat:sgtin-96:0.0614141.112345.\*” would otherwise result in 274,877,906,943 EPC instances being returned.

### **Section 9.6.4: AssocTableEntry**

The “key” for an Association Table Entry is an EPC value, but the ALE specification is silent on the format of the EPC value. Intermec ALE attempts to convert EPC values read from tags first into “epc-tag” format, and then (if that fails due to the tag not containing a valid EPC) into “epc-hex” format. No other formats are supported. Therefore, only “epc-tag” and “epc-hex” values should be specified as keys in an AssocTableEntry element.

It is possible to provide “epc-tag” and “epc-hex” key values that represent the same EPC value. In that case, the “epc-hex” key is never used since the “epc-tag” format match is always attempted first.

### **Section 9.7.2: RNGSpec**

The ALE specification is silent on the behavior of the random number generator. Intermec ALE provides a pseudo random number generator (PRNG) that is seeded with the system time. The PRNG is not cryptographically secure and should not be used to encrypt information.

### **Section 10.3.2: Logical Reader API Conformance Requirements**

This section of the spec lists the required behavior with regard to composite, externally-defined, and API-defined readers.

Intermec ALE only allows the creation and modification of composite readers. Externally-defined readers are defined in the “ale.conf” configuration file. Intermec ALE does not support creating base readers through the API. Modification of externally-defined readers is limited to adding/updating/changing the properties associated with those readers. No creation or deletion of those readers is supported.

### **Section 10.6: Tag Smoothing**

Tag Smoothing is optional. Intermec ALE does not implement tag smoothing.

### **Section 11.8: Access Control API Partial Implementations**

As explained in this section, a minimal Access Control API implementation consists of implementing three of the methods (getStandardVersion(), getVendorVersion(), and getSupportedOperations()). All other methods can return a UnsupportedOperationException.

Intermec ALE implements a minimal Access Control API by implementing only the getStandardVersion(), getVendorVersion(), and getSupportedOperations() methods. All other methods return an UnsupportedOperationException.





**Worldwide Headquarters**

6001 36th Avenue West  
Everett, Washington 98203  
U.S.A.

**tel** 425.348.2600

**fax** 425.355.9551

[www.intermec.com](http://www.intermec.com)

IF61 ALE Engine User's Guide



P/N 934-026-001